International 99/4 Users-Group

EDUCATION • ENTERTAINMENT • PERSONAL ENRICHMENT

NEWSLETTER

Vol. II No. 6 | Oklahoma City, Oklahoma | November 15, 1982

TI-FEST FUN FOR ALL

The first ever TI-Fest sponsored by 99er Magazine was held in Brooks Hall, San Francisco the weekend of Oct. 29, 30, & 31. Thousands of those who attended were able to see the latest hardware and software products offered by Texas Instruments, and gain "hands-on experience" by using many of the over one hundred 99/4A systems that were on display.

Creative workshops on a multitude of subjects including Forth, Pascal and Assembly Language were well attended and quite informative. These workshops, conducted by TI employees and staff from 99er Magazine, provided a wealth of information for both the novice computer owner and the experienced computer guru. A two hour question and answer session, conducted by Mr. Don Bynum, Vice President of TI's Consumer Products Division, was the highlight of these seminars and gave 99/4 owners an opportunity to not only air their grievances, but to delve into the future plans of the Personal Computer Division.

Several third party software manufacturers were in attendance showing off their latest software products and gaming competition sponsored by TI and 99er Magazine kept the crowd moving rapidly around the assembly hall.

TI-Fest not only had local appeal to San Franciscans, but also brought in 99/4 owners from such exotic places as England, Holland, Canada, Germany, and the Phillipines. Many 99/4 owners from various states in this country attended as well, and local Users-Groups from Houston, Portland, Los Angeles, Orange County (CA), Washington D.C. and others all had representatives in attendance.

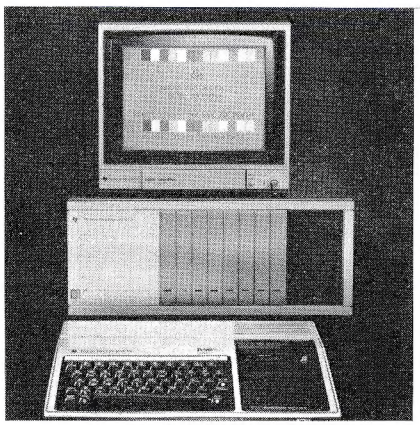
Most 99/4 users seem to find real interest in the exciting new game module Parsec. Hundreds of 99/4 owners at the International Users-Group booth gathered to see the Tandon dual-sided disk drive put thru its paces. With the new Disk Manager II which TI loaned us for the occasion, we were the first at the TI-Fest to show you truly can initialize all 718 sectors of a dual-sided disk.

The editor of 99er Magazine, Gary Kaplan, along with his staff and the dozen or so TI employees from both Lubbock and Dallas, are to be congratulated for putting on a great show for all of us who were fortunate enough to attend. We here at the Users-Group hope that this was only the first of many TI-Fests, which we would like to see held in various parts of the country in the future.

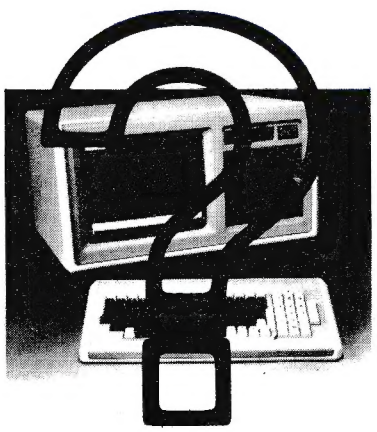
Who Knows For Sure?

Hand-Held Computer

99/4A $199.00



64K Computer



RUMORS ON NEW COMPUTER

For the past year, rumors about a new personal computer from Texas Instruments have been circulating in the computer industry. Several prototype models have been built by TI and some have even been shown to selective consumers in various parts of the country.

Is a new computer from TI rumor or a fact? At this time only TI knows for sure. What we do know is that TI has been working on a new console for some time now, that incorporates several of the more desired features that to this time have only been options. A prototype model called the "Ranger" featured built-in speech, 64K of internal RAM, built-in RS232, and featured a version of Extended Basic as the resident language. A great machine for certain segments of the consumer marketplace, but still not all things to all people.

Other rumors say that TI has been working on a lower end computer to compete with the Sinclair and Timex models. This also makes sense to us as we see a computer selling in the $100 price range having a great deal of appeal to a wider market segment. With TI's recent announcement that they will not build their TI-88 calculator, a low end computer may be in the offing.

Our sources at TI will not give us any indication of what to expect other than to say that there will be a few surprises at the January Consumer Electronic Show.

Our guess (and that's all it is at this point) is that TI will indeed introduce two new computers in January. A hand held unit with 4K of RAM that will be expandable to 16K, and a higher end model with 64K of RAM. The hand held unit should hit the market with a suggested retail price of around $100 and the upper end unit may well be in the $500-$700 range. This type of move would give TI a broad range of appeal to the entire personal computing population. We also think that the 99/4A as we know it today will continue to be around for a long time to come; however we feel that the retail price may fall once again to $199 and stabilize at that point.

Additionally, TI is expected to announce a new Command Module which will feature the familiar E.T. space creature. The new E.T. Command Module is due to be ready some time in the second quarter of 1983.

POTPOURRI

ADDITIONAL LINES

Have you ever wished that you could use more than 4 lines in a program statement? Well you can!

A flaw in the basic language of the 99/4A console will allow more than 4 program lines by using the following editing trick. Let's say that you want to print this statement on your screen; Mary had a little lamb, its fleece was white as snow and every where that Mary went the lamb was sure to go.

By typing in

```
10 PRINT" Mary had a little
lamb its fleece was white as
snow and every where that M
ary went the lamb was sure "
```

You will notice that the computer will except nothing past the last ". However if you edit the line by typing 10 and hitting a FCTN X and then skip to the end of the line you can continue to type in characters until the end of the fifth line.

Should you still need additional characters you can do the same thing for a sixth line. Try it yourself and see if it will be of help in your programming.

WHY CAN'T TI?

Why can't TI come up with what we have just seen from their friendly competitors at Heath/Zenith?

What we are referring to is excellent audio-visual presentation of the fundamentals of programming in Pascal. The course entitled, "Programming in Pascal, an individual learning course" is a totally wonderful work of art that consists of a 510-page manual and 5 cassette tapes.

The course consists of several lessons which break the material into easily digested parts. The pages of the manual show more than mere words; they incorporate overhead projections and flip charts that the user can easily follow and give even the novice Pascal programmer a sense of accomplishment early in the courseware.

Heath/Zenith is to be applauded for this effort and the courseware is a must for anyone who is seriously interested in learning Pascal.

For full details on pricing and availability contact your nearest Heathkit store or Heath/Zenith Data Systems dealer.

RUN "CS1"

From the Washington DC Users-Group comes this interesting bit of information. Extended Basic contains an undocumented, but implemented, feature of interest to all users of cassette tape. RUN "CS1" can be used either as a command or a statement. By using it as a statement such as: 560 RUN "CS1" programs can be chained, i.e. one program can call another. The screen will give the normal instruction for loading a tape and when the program is loaded, the computer will automatically start running the program.

This command is the equivalent to the commands of:

```
>OLD CS1
>RUN
```

but it is now under program control.

CALL LOAD

The statement CALL LOAD (-31878,0) **does not** disable the sprite motion in Version 110 of TI's Extended Basic, and **will not** increase the operating speed of a program as it did in Version 100. We have asked if there is a statement that will disable the sprite motion in Version 110 and have been unable to get a straight answer at this point.

MEMORY STATUS

Need to know how much memory you have used in a Basic program? Try this little 2 line routine for the answer.

```
>1 A=A+1
>2 GOSUB 1
>RUN
```

The computer will run until the memory is full, at which time a MEMORY FULL IN 1 message will appear. At that point type in >PRINT A and the number that appears will be the amount of memory remaining for programming.

TE-II OUTPUT FILES

For those of you who use the TI Terminal Emulator II and save data to an external device such as a cassette or disk here is an Extended Basic program that will allow you to retrieve the data you have saved.

```
100 INPUT "device and file name"
  :FILE$
110 OPEN #1:FILE$,INPUT,
  DISPLAY,FIXED 80
120 LINPUT:A$
130 IF EOF(1) THEN 160
140 PRINT A$
150 GOTO 120
160 CLOSE #1
170 END
```

RIGHT JUSTIFY NUMBERS

One of the problems of TI Basic is that it does not automatically right hand justify columns of numbers either for screen or hard copy print outs. This is a simple program to solve by using the following subroutine in your programs.

```
1000 REM RIGHT SUBPROGRAM
1010 IF POS(STR$(RJN),".",1)
  THEN 1040
1020 PRINT TAB(T-LEN(STR$
  (RJN)));STR$(RJN)
1030 RETURN
1040 PRINT TAB(T+1-POS(STR$
  (RJN),".",1));STR$(RJN)
1050 RETURN
```

Now anytime in the body of the program you wish to print a right justified number (specified or calculated) simply let RJN equal

that number. Let T equal the column to contain the decimal, and GOSUB 1000.

Example:

```
100 CALL CLEAR
110 T=11
120 INPUT "ENTER A ":A
130 CALL CLEAR
140 RJN=2.4*A-1
150 GOSUM 1000
160 RJN=192
170 GOSUM 1000
180 T=25
190 RJN=A+11
200 GOSUB 1000
210 INPUT "ENTER X ":X
220 RJN=X
230 GOSUB 1000
240 INPUT "PRESS ENTER TO
        CONT. ":Z$
250 GOTO 100
```

SOFTWARE EXCHANGE LIBRARY

Due to our ever increasing membership we feel it is in our best interest to once again explain the workings of our Owner Written & Translated Software Library.

This Library is open to all members of the International 99/4 Users-Group. *All program submitted to the Library remain the property of the original member who coded or translated the program and submitted it to the exchange. The International 99/4 Users-Group Inc. does not claim any proprietary rights to any program in the Library and can not be held responsible for their contents.

Many of the programs in our Library do not contain operational instructions and the user may experience some difficulty in their operation. **Our charter will not allow us** at the International 99/4 **to make any changes to any program submitted to us,** so we encourage our members to correct errors that they may find and resubmit programs to us. We do screen each program before it is entered into the Library however from time to time programs with "bugs" may be found in the Library. We again encourage any member who finds mistakes or "bugs" in these programs to correct them and resubmit the program in a corrected version.

The Exchange policy is as follows:

Any Member who submits a workable program (on cassette tape or disk) written in TI Basic, TI Extended Basic, TI LOGO or TMS 9900 Assembly Code, which will operate on the Texas Instruments 99/4 or 99/4A, may choose any four (4) programs from the Exchange Library. Your original tape or disk will be returned to you with your selections.

When submitting programs to the Exchange please type in a REM statement with your Name, City and Zip Code for each program submitted.

Members who do not program or do not wish to participate in the Exchange program may purchase any program in the Library for $3.00 each either on cassette tape or 5 1/4" floppy disk. This charge is to cover the cost of the media and our cost of operation and mailing. (Multiple programs are placed on both tape and disk.)

All orders for tape or disk based programs must meet a total minimum order charge of $12.00.

Hardcopy (program listings) may be ordered for $1.25 each and there is no minimum order charge.

We are unable to accept the following type of programs for our exchange:

1. Programs that duplicate existing programs in our library. Except for revisions of existing programs.
2. Programs that are published by 99'er magazine.
3. Programs that were originally written in a TI Basic language and published under a copywrite.
4. Program titles which may infringe on a copywirte i.e. E.T., Starwars, Munchman, etc.

*Programs submitted to the Library and listed in the Exchange Library catalog constitute a legal copywrite for the original coder or translator of these programs. Any unauthorized duplication or resale of any program listed in the Owner Written & Translated Software catalog is a direct violation of Federal law.

THE ASSEMBLY LINE

ASSEMBLY LANGUAGE BLUES: HOW TO USE YOUR MINI-MEMORY LINE BY LINE ASSEMBLER

I've owned my 99/4A since November of 1981. Since then I've been disappointed by it's speed, or more properly, the lack of it. One of my primary reasons for choosing it over the ATARI 400 or VIC-20 was because it was based on a 16 bit microprocessor rather than the 8 bit system the other two use. A computer buff friend had told me the 16 bit CPU should theoretically run four times faster than an 8 bit CPU. I didn't understand why the speed was quadrupled rather than doubled, but it sounded good. Theories often sound real good.

I bought my 99/4A sight unseen, a pig in a poke, it was "oinking" fifteen minutes after set-up. I keyed in this program:

```
1 FOR X = 1 TO 10000
2 NEXT X
```

I broke out my stopwatch and readied my finger, wondering if I could push the button on and off fast enough to record the event. I pushed 'enter' and hit the start button on the watch. I watched the screen for the word 'DONE' like an Olympic sprinter listens for the bark of the starting pistol. I waited....The program ended 28.9 seconds later, and I groaned. I could've had a V-8, or at least gotten one out of the fridge, while waiting for the program to finish. From that point on, my dreams of ultra-fast computing speed rested on an esoteric word I had also heard from my friend -assembly language.

I eagerly awaited TI's release of their Editor/Assembler package. Again TI caused me to groan when I discovered you had to have a disk drive system and the 32K Memory Expansion to run assembly language. A quick calculation later and I realized I would have to fork out funds in the neighborhood of $1000 if I really wanted to run assembler. That's a pretty rich community considering my rent is only $180 a month. Since I bought my computer for hobby purposes,

"For less than $100 I could unleash the full power of my computer by writing and running assembly language programs."

and not for practical reasons, I just couldn't justify spending a grand on a computer language when I don't even own a color tv.

Mini-Memory solves the problem. TI answered my prayers when they came out with this. For less than $100 I could unleash the full power of my computer by writing and running assembly language programs. The Mini-Memory is much more than an assembler. TI threw that in as a bonus. Primarily it was designed as a nonvolatile 4K memory addition. Basic programs can be loaded in seconds if stored in the Mini-Memory. The Mini-Memory can just as easily be used for data file handling. Anyone who ever used a cassette tape data file will fully appreciate this feature. But now I'm digressing from the subject at hand. Let me summarize by saying the Mini-Memory is real value for the money.

First off, this is not going to be a technical dissertation. I don't really know the inner workings of the TMS9900 CPU chip in your computer. I do know enough about 9900 assembly language as used in the Mini-Memory to get you started in assembler programming. I will take you step by step, starting out very simply with short programs. You can load and run the simple programs in about 5 minutes. They may be trivial examples, but the lessons to be learned are immense. I whooped and hollered when I first figured out how to get the program to display characters on the screen.

Needless to say, writing an assembly program is no way near as easy as writing in BASIC. Why, then, would you want to write in assembly language anyway? Speed and computational power. The gain in speed is phenomenal, as I will demonstrate.

I'll guarantee that unless you already know something about 9900 assembly language programming, you won't be able to do much with the documentation that comes with the Mini-Memory. It does advise you to order the Editor/Assembler manual from TI. I would have been lost if Charlie at the International Users-Group hadn't lent me his copy. When I first opened the Editor/Assembler manual, I thought Charlie had forgotten to give me volume one. Big surprise: there is no volume one. The headache was just beginning. The Editor/Assembler manual was not written to teach you how to write in assembly. It is strictly a technical manual for the Editor/Assembler system. Still, you can't get by without it, so I took two aspirin and started wading. The best advice I can give you about the manual is to read the appendicies first. They do provide some background knowledge. Also, skim the glossary before reading the

main text.

Let's start learning to use assembly language. Put your Mini-Memory into your computer and load the line by line assembler tape that comes with it. Return to the master selection list and choose the Mini-Memory option. When the Mini-Memory list appears, press 2. Type in NEW, and press enter. Now you can begin to enter assembly language instructions. Unfortunately, the Mini-Memory documentation doesn't even give you a list of the instruction set for the TMS9900 chip. I had to get those from the Editor/Assembler manual. Press enter eleven times. What you are seeing is part of the coding for the 'LINES' program that loads along with the line by line assembler. The first column is the hexadecimal address of a word of memory in Mini-Memory RAM. The second column is the instruction or data that is stored at that address. Look at address >7D14. (The ">" symbol indicates a hex number.) It contains "0203".

This is the machine language form of the assembly language instruction mnenomic "LI 3". Since this is a two word instruction we also need to look at the next word address, >7D16. It contains "8000". The complete instruction encoded by these two memory words is "LI 2,>8000". Try it and see. Press the space bar (SB) AORG (SB)>7D14 and press enter. This repositions the program to allow us to re-enter address >7D14. Now type "(SB) LI 3,>8000", and enter. >7D14 and >7D16 didn't change because this is the identical instruction that they already held. Reposition the program back to that instruction by using "(SB) AORG >7D14". Type in (SB)LI 4,8000. Note that both words changed, but >7D16 encoded as "1F40" and not "8000". This happened because when you don't precede a number with ">" the assembler assumes you are entering a decimal (base 10) number. Since the computer can only work with hexadecimal (base 16, the assembler converts all decimal numbers to hex. This saves you from having to do the tedious task. Now enter "(SB)AORG >7D14" and re-enter the instruction as "(SB) LI 3,32768". Since 32768 is 8000 in hex, we have restored words >7D14 & >7D16 to their original state. Even if you change the instructions back to what they should be, 'Lines' will no longer run. It seems that whenever you enter the assembler, you change something in the 'Lines' program. My guess is that the assembler uses some of the same memory space as 'Lines'. Not to worry, as you can always reload it from the tape. Since we already mucked up 'Lines', let's write a program that will zeroize a portion of memory so we can have a clear space for other programs we will write. This isn't necessary, as we could simply write over the 'Lines' coding, but this way we won't confuse our coding with the previous program.

We will call the program 'Clears', and we will place it at the bottom of the free memory space so we can save it and still have plenty of room for our other programs. This is the first assembly program I ever wrote, and to me it was a milestone. It is only eight words long (16 bytes) and its function is simple to understand.

First, reposition the assembly starting point by typing "(SB) AORG >7FD0" and entering. I will use "b" now instead of "(SB)" to indicate a space. This is the convention the Mini-Memory documentation uses. The program is only five lines long:

```
b LI 1,>7D00
L b CLR *1+
b CI 1,>7FD0
b JNE L
b B @>71AC
```

After entering the program, it will appear on your screen like this:

```
7FD0 0201    LI 1,>7D00
7FD2 7D00
7FD4 04F1    L CLR *1+
7FD6 0281    CI 1,>7FD0
7FD8 7FD0
7FDA 16FC    JNE L
7FDC 0460    B @>71AC
```

The 1st, 3rd, and 5th instructions require two words of memory; the other two instructions need only one word. For the benefit of those who don't have the Editor/Assembler manual, I'll briefly explain each of the instructions.

LI 1,>7D00

"LI" is the symbol for the load immediate instruction. >7D00 will be loaded in register 1. Sixteen registers (0-15) are available for your use. Think of them as pocket calculator memories in which you can store hex values of 0000-FFFF. >7D00 is the first address we wish to clear.

L CLR *1+

"CLR" stands for clear and will place "0000" in whatever operand follows it. Here the operand is "*1+". "1" again refers to reg #1. The "*" stands for indirect. The CLR instruc-

tion will not zeroize reg #1, but instead will zero the address stored in #1. Therfore, >7D00 will contain "0000", but reg #1 will be untouched. The "+ tells the

"It would be less confusing if they would have called it "The Address of the First Reference Statement."

computer to automatically increment reg #1 after the rest of the instruction is completed. So, #1 will now contain >7D02, which is the next address we want to zero out. Let me illustrate the effect of this instruction:

Before: reg #1 = 7D00
Address 7D00 = doesn't matter
After: reg #1 = 7D02 Address 7D00 = 0000

CI 1,>7FD0

"CI" means compare immediate. The content of reg #1 is compared to the constant >7FD0, not the contents of *address* >7FD0. We want the program to quit zeroing addresses when it reaches >7FD0, because it contains the first instruction of this program.

JNE L

Jump if not equal. If the previous comparison wasn't equal, that is if reg #1 didn't contain >7FD0, the program will loop back to the address which contains label "L", which happens to be >7FD4.

B @>71AC

Branch to address >71AC. This ends our program, and returns control back to the assembler. >71AC is the start of the assembler program. You can think of the branch instruction as being a basic language GOTO.

Go ahead and enter the program, but you must do one more thing before you can run it. We have to tell the computer where program 'Clears' begins. We do this by making an entry in the "REF/DEF" table. Page 18 of the Line by Line assembler booklet tells you how to do this, but just in case you didn't understand it, I'll help you with this one. Address >701E is used by the computer to store the address of what TI calls "The Last Free Address in Memory". It would be less confusing if they would have called it "The Address of the First Reference Statement", because that's what it is. Type in "b AORG >701E", and let's see what it is. You will see it contains >7FE8. That happens to be the first address of the 'Lines' reference statement. Each REF occupies 8 bytes; six are for the name of the program, and the last two contain its first address. Since the REF/DEF table runs backwards in memory, we subtract 8 from >7FE8 to figure out what address to start our REF for 'Clears' at. >7FE8 -8 = 7FE0. We must place this address in >701E, which is where we happen to be. We use the "DATA" command to do this. Type in and enter "b DATA >7FE0". Now enter "b AORG >7FE0", so you

Scott, Foresman

Combining computers and classrooms...with confidence

"Scott, Foresman's expertise and commitment shows through in their microcomputer courseware...This is the way software should be; easy to use, effective and reliable..."

Dr. D. Thomas King, Assistant Director, St. Paul Public Schools

It takes more than knowledge to provide effective, useful computer products. It takes a commitment. A commitment to education—providing the skills and concepts students need to learn; a commitment to expertise—making materials *useful* to teachers; a commitment to simplicity—providing equipment that's easy to use; a commitment to students and educators alike—resulting in friendly, effective, reliable computer products.

Discover the Scott, Foresman commitment. Send for our **FREE** catalog and compare.

Electronic Publishing Division
We teach students in ways you may have never imagined.

Scott, Foresman and Company
1900 East Lake Avenue • Glenview, Illinois 60025

D99/10

Please send me a **FREE** Scott, Foresman Electronic Publishing catalog. I am interested in software for:

☐ School Management Applications ☐ Mathematics
☐ Reading ☐ Science ☐ Computer Literacy

Microcomputer in use:_____.

☐ I am interested in microcomputer equipment.
Please have a representative contact me.

Name_____

Position_____

School_____

School Address_____

City/State/ZIP_____

can enter the REF statement in the proper place. We decided to name the program 'Clears', so we will need to know the hex symbols for each of the letters. Don't know what they are? No problem, the assembler will do it for you by using the text command. Enter "bTEXTb' CLEARS' ". Again, do not type the *double* quotes, but you must have the *single* quotes. Three words of memory will be filled, so you should now be at >7FE6. This is where we enter the starting address of 'CLEARS'. Enter "bDATA >7FD0".

While we're in the REF/DEF table, let's change the old 'LINES' REF so we can use it for our next program. We should be at address >7FE8, so we can simply over write 'LINES'. Our next program will begin at >7D00. Let's call it "MYPROG". Enter "bTEXTb'MYPROG' ", then enter "bDATA >7D00".

The work is all done, so let's run 'CLEARS'. To exit the assembler, enter "bEND". Now press enter until the Mini-Memory selection list appears. Press "2" to run. When it prompts you for the program name, type in CLEARS and enter. You'll be surprised at how fast it executes. Now for the speed improvement demo I promised you.

We want to write the equivilent of the basic 1-10000 loop. Now that we have the general idea of how the assembler works, it won't be as difficult:

```
7D00 0000 b A0RG >7D00
7D00 0201 b LI 1, 10000
7D02 2710
7D04 0601 L b DEC 1 (decrements
    reg #1 by 1 and compares it to
    zero)
7D06 16FE b JNE L (loops to L until
    reg #1 contains zero)
7D08 0460 b B @>71AC (ends
    program with branch back to
    the assembler)
7D0A 71AC
7D0C 0000 b END (exit the
    assembler)
```

Now run 'MYPROG' using Mini-Memory selection 2. *We finally have that elusive ultra-fast speed!* The program completes in the twinkle of an eye. I couldn't work the stopwatch fast enough. Let's expand the program to count down from 1000000: You can run 'CLEARS' to erase the old code, or write over it.

Since 1000000 is too large to store in one word, we will have to "cascade" two regs.

```
    LI 1, 1000
L   LI 2, 1000
D   DEC 2
    JNE D
    DEC 1
    JNE L
    B @>71AC
```

This program clocked in at 12 seconds, which means it's executing about 200,000 instructions each second. In case you ever wondered, the clock speed of your 99/4 is 3.579545 megahertz. If you want to compare this with an 8 bit microprocessor, you must remember we are dealing with words, but they only deal in bytes. The big advantage of the 16 bit CPU comes in the math department. The machine language instruction set of the TMS9900 includes multiply and divide, not so with an 8 bit CPU, which only has add and subtract to work with. They must execute a myriad of machine instruction to do what the TMS9900 does with a single instruction, and even then they take shortcuts. If you do the basic statement PRINT 1/3*3 on an ATARI, you will come up with .9999999999 for an answer. On the 99/4 you will get 1.

Now that you are able to run the simple program examples, we'll increase the program complexity a bit. The next program will demonstrate how to use the speed of assembly language to change your monitor screen faster than 1000 times per second. This program will display a fast eight digit counter:

```
LI 0,>187 (first screen
    position of counter)
LI 1,Z (address of first counter
    character)
LI 2,17 (number of bytes to display
    on screen)
LI 4,>3A3A
S BLWP @>6028 (branch to screen
    write utility)
```

```
LI 3,14
I INC @Z(3)
C 4,@Z(3) (compare reg 4 with
     address to z & value of R3)
JNE S
MOV @Z, @Z(3) (copy the value
     at z into address Z & reg 3)
DECT 3 (decrement reg 3 by 2)
JMP I
Z TEXT ':0:0:0:0:0:0:0:0:0'
```

Again, if you load this program starting at address >7D00, you can run it by entering 'MYPROG' when the Mini-Memory run option prompts you for the program name.

Getting characters displayed on the screen was my biggest problem. It appears that the Mini-Memory command module booklet is incorrect. I stumbled over this for hours before I realized what was wrong: THE STANDARD UTILITY REFERENCES WILL NOT WORK. I was flabbergasted when the booklet examples didn't work. As an example, on page 35 the VSBW routine is described. This utility should let you display a single character on the screen. The example the booklet gives is:

```
LI R0,>0200
LI R1,>4100
BLWP @VSBW
```

The example did not work. It seems that the reference "VSBW" is not defined in the Mini-Memory ROM. Perhaps this was a feature that TI had intended to add but never did. The solution is easy: you use the actual address instead of the reference BLWP @>6024. Luckily, the reference addresses are given in the Mini-Memory booklet. With this bit of knowledge we can now use the utility routines.

The utilities are subprograms that save you the bother of having to write coding for common procedures. In the counter program "BLWP @>6028", accesses the utility routines.

The utilities are subprograms that save you the bother of having to write coding for common procedures. In the counter program "BLWP @>6028", accesses the utility which allows multiple characters to be displayed on the screen. "BLWP" is the mnemonic for 'branch and load workspace pointer'. We don't use the simple branch instruction because we want to preserve our own workspace registers.

By the way, the counter program is an endless loop. It is also uninterruptable. You will have to turn your console off when you tire of watching it count.

Since speed is the forte of assembly programming, I set myself a goal to improve the speed of the counter program by at least 20%. My first revision missed this goal by 3%, but I improved on this program dramatically by working on a byte level rather than a word level. The resulting program counts to 10000 in only 3.8 seconds; about half the time required by the first program.

Here is the improved counter listing:

```
JMP B
T TEXT '00000000'
B LI 0,>18C
LI 2,6
LI 1,T+2
LI 4,>3A00
D LI 3,6
BLWP @>6028
I INC @T(3)
CB @T+1(3),4 (compare bytes)
JNE D
MOVB @T,@T+1(3) (move bytes)
SWPB @T(3) (swap bytes)
INC @T(3)
CB @T+1(3),4
SWPB @T(3)
JNE D
MOVB @T,@T(3)
DECT 3
JMP I
```

At this point let me issue a challenge to the more advanced programmers who are reading this. Can you improve the counting speed by another 20%? I have an untried idea in mind as to how this might be done, but it would lengthen the program quite a bit, and I doubt if the improvement would be that great. The solution would be a 6 digit-on screen counter that increments by ones to 10000 in about 2 1/2 seconds.

I hope to have a column such as this in every issue, and I would like your comments, ideas, and requests. To save a lot of words, I will assume you have access to an Editor/Assembler manual or some other manual of 9900 assembly language programming. I am not a professional programmer, just a hobbyist like most of you. I will pass on what I discover about the 99/4, and gladly relay any of your discoveries. Feel free to write or call me at home:

Bill Gronos
9505 1/2 S.E. 15th #B
Midwest City, OK 73130
Phone (405) 733-9736

In my next column I will show you how to use assembly subroutines in TI BASIC. We will build on the counter program and use it in a BASIC program where the ultra-fast speed is required. Hopefully, I will gain access to a full Editor Assembler system so I can help those of you who now have it on a shelf collecting dust.

For Mini-Memory owners who fear they may lose the sprite capability of the Extended Basic module by using the console module slot for the Mini-Memory, here is a short Basic program to give you an idea for using sprites with the Mini-Memory inserted:

```
100 CALL CLEAR
110 CALLPOKEV
     (768,98,128,161,1,208)
120 CALL POKEV(1920,50,50)
130 CALL LOAD(-31878,1)
140 GOTO 140
```

This will be explained in a future column.

WHAT IS PASCAL?

This is a question we here at the Users-Group hear almost every day. Pascal is a high level programming language (software) that was developed in the late sixties by Professor Niklaus Wirth at the Swiss Federal Technical Institute. He wanted to create a language suitable for teaching the concepts of disciplined, structured programming.

Over the past decade further refinments in the Pascal language have been made in both this country and in Europe. One of the foremost versions of Pascal was adopted for use by TI on its Home Computer, version 4.0 from the University of California San Diego.

Pascal is not hard to learn, but don't be mislead; the average basic programmer must change the entire way he thinks about programming in order to master it. There is a lot of disk shuttling also due to the programming tools (Editor, Filler, Compiler, Assembler and Linker) which are needed to write a Pascal program.

Pascal is not extremely fast, only 3 to 5 times faster than console basic, but the transportability to other systems with little or no changes in code makes it a highly desirable language.

To run Pascal on the 99/4 or 99/4A you must have a 32K Memory Expansion, and either the p-Code card or old style p-Code box peripheral as well as a cassette recorder or disk system. To write programs in Pascal on the TI equipment you must have a disk system, 32K Memory Expansion and p-Code card or box as well as the p-Code software programming tools mentioned above.

DISK MGR. II

TI was kind enough to give us an advance copy of their soon to be released Disk Manager II Command Module for use at TI-Fest. Our test of the new module shows that it will indeed initialize both sides of a dual sided disk (718 sectors) when using the Tandon dual sided disk drive. Although the Disk Manager II does have a prompt for double density it is inoperative which leads us to believe that some day there may just be a Disk Manager III.

As of the writing of this newsletter no decision has been made by TI on what type of exchange or sale arrangement will be made for those of you who wish to own this Module.

FORTH IS COMING

Shown for the first time at TI-Fest, TI's version of Forth is to become available during the first quarter of 1983.

Forth was invented by Charles Moore and has attracted quite a following among many people in the computer industry including hardware designers and programmers. It is a relative small language that is very powerful, and it encourages the programmer to expand it to fit his own needs. This point was well brought out by some of the sample programs that were seen at TI-Fest.

I was amazed at the speed in which programs executed while using Forth. A musical program (Bach) which was written by a TI employee gathered large crowds at our booth while it was playing.

From what we saw it seems that Forth requires a great deal of attention on the part of the programmer, somewhat less perhaps than machine code but certainly more than basic. The syntax of Forth can become quite confusing when a legal statement can go something like this >:?@@.;

One nice thing about Forth is that there is a wealth of written information on its operation (that is if TI has not changed it too much) which should be helpful to the Forth Novice.

No prices or availability schedule was released.

SCHOLASTIC SPELLING

A complete spelling program has been created for TI by Scholastic, Inc. A separate command module exists for each of nine levels. The length of the word lists, as well as the difficulty of the words, increases as one progresses from one level to the next. Each module consists of thirty-six word lists or lessons. The words in each lesson are grouped according to vowel sounds and by similarities in structure. Every sixth lesson is a review containing words from the previous five lessons. A student workbook provides a printed list of the words and a written activity for each lesson.

After the proper level has been determined, the student may select the lesson or word list he wishes to study. The words in that lesson are then printed on the screen one at a time as they are being pronounced aloud by means of the speech synthe-

sizer. One may notice at this time that the quality of the spoken word is poor. Hopefully, advancements in technology will allow for future improvement in this area. Meanwhile, it must be kept in mind that the initial hearing of the word was accompanied by a visual printing. If a student proceeds with the activities in the order that is suggested, he will be so familiar with the words in a particular lesson, that when he arrives at the point where the only clue is the spoken word, he will have heard that sound many times while seeing the word in print. Certainly, the many positive features outweigh this one negative one.

After the student has familiarized himself with the words in the lesson, he is given an opportunity to select any one of three computer games designed to give him practice in spelling skills.

"Spelling Bee" is a drill and practice game in which the student selects from among these four clue options:

1. All clues
2. Vowel clues
3. Consonant clues
4. No clues

For each word, the clues are given, the word is pronounced, and then the student spells the word. For the greatest learning opportunity, the game should be played four times, choosing each clue option in the order they are listed. This will allow much repetition with diminishing clues until, at option 4, the student is spelling the word with no clues except blanks corresponding with the number of letters in the word. "Spelling Bee" is for one player.

"That Did It" is a game that should be played only after the student has become thoroughly familiar with the list words. This, of course, is the purpose of the "Spelling Bee" game, which ideally should precede all other activities. By playing "That Did It", the student can discover spelling patterns, arrangement of vowels and consonants, and patterns in the structure of words. The student is shown the number of blanks corresponding to the number of letters in one of the list words. To play the game, he selects one letter at a time which he thinks might be needed to spell the unknown word. If the letter he has selected is used to spell the word, it will appear on the screen in all its proper blanks. If the letter selected does not belong in the word, one of seven triangles will fall into a magician's hat, signifying a miss. The student continues until he has spelled the word or has used up his seven incorrect letter selections. As a student gains experience, he is likely to learn to make wiser selections. If he is working on a lesson using words with long "e" sounds, he may learn that the letter "e" is a very good selection. "That Did It" is for one or two players.

"Space Race" is an ideal game for evaluating the learning made possible by the other two games. Each word is pronounced and the student spells it with no clues. The student is taking a traditional spelling test in a game format. One or two players may participate.

In each game, personalized affirmative responses are given for correctly spelled words. Incorrectly spelled words do not receive a negative response. The correctly spelled word is simply flashed on the screen several times. At the conclusion of each game, the student receives this message, "Phyllis, you spelled these words correctly." The words are then printed on the screen as they are being pronounced. If some words were incorrectly spelled, the player is told, "Take a look at the words you need to learn, Phyllis." After those words have been shown to the student, he is invited to play the game again using only the missed words.

Scholastic Spelling could easily be the entire spelling curriculum for a student spending 15-30 minutes daily playing the computer

SOFTWARE

games. One lesson a week would allow a student to finish a module or level in a school year with approximately thirty-six weeks.

A classroom teacher might decide that the value of the program would be the meeting of needs of some 1/3 of the students in a typical classroom. This group would include those who would have difficulty mastering the "grade level" material, as well as those who need a further challenge because they have already mastered this year's textbook.

HOLIDAY SCHEDULE

The International 99/4 Users-Group offices will be closed for the Thanksgiving and Christmas Holidays during the following days:

November 25 & 26 (Thanksgiving)
December 24, TO January 3, 1983
(Christmas and New Years)

PARSEC

TI's newest Arcade Entertainment Module is FANTASTIC.

Fly into combat with the starship PARSEC. Destroy rebel alien fighters and cruisers by outmaneuvering them and laying down withering fire from your lasers. Then try to survive the deadly asteroid belt.

Parsec pushes the 99/4A console to its limits, with swift moving graphics and totally new digitized speech patterns. The fast paced action will keep many a player up all night trying to reach their maximum scoring level with Parsec's multi-screens action.

In our opinion Parsec is the finest Module that TI has produced todate and offers true arcade style action.

Parsec does require the 99/4A console and joysticks are suggested. Use of the Speech Synthesizer is optional.

Users-Group Rating ****

International 99/4 Users-Group

EDUCATION • ENTERTAINMENT • PERSONAL ENRICHMENT

P.O. Box 67
Bethany, OK
73008

BULK RATE
U.S. POSTAGE
PAID
Permit No. 56
Bethany, OK

RETURN POSTAGE GUARANTEED

SKIP HARRISON
RT. EAST BOX 99
NEWPORT, OR 97365